PENJADWALAN MATAKULIAH DENGAN MENGGUNAKAN ALGORITMA GENETIKA DAN METODE *CONSTRAINT SATISFACTION*

Joko Lianto Buliali[1] Darlis Herumurti[2] Giri Wiriapradja[3]

[1,2,3]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Email: [1]joko@its-sby.edu, [2]darlis@its-sby.edu, [3]giri03@inf.its-sby.edu

Course scheduling problem has gained attention from many researchers. A number of methods have been produced to get optimum schedule. Classical definition of course scheduling cannot fulfill the special needs of lecture scheduling in universities, therefore several additional rules have to be added to this problem. Lecture scheduling is computationally NP-hard problem, therefore a number of researches apply heuristic methods to do automation to this problem.
This research applied Genetic Algorithm combined with Constraint Satisfaction Problem, with chromosomes generated by Genetic Algorithm processed by Constraint Satisfaction Problem. By using this combination, constraints in lecture scheduling that must be fulfilled can be guaranteed not violated. This will make heuristic process in Genetic Algorithm focused and make the entire process more efficient. The case study is the case in Informatics Department, Faculty of Information Technology, ITS.
From the analysis of testing results, it is concluded that the system can handle specific requested time slot for a lecture, that the system can process all the offered lectures, and that the system can produce schedules without violating the given constraints. It is also seen that Genetic Algorithm in the system has done optimation in finding the minimum student waiting time between lectures.

Kata Kunci: *genetic algorithm, constraint satisfaction problem, optimation, lecture scheduling*

1 PENDAHULUAN

Permasalahan penjadwalan untuk pengajaran mendapatkan perhatian dari banyak peneliti. Sejumlah metode telah dihasilkan untuk mendapatkan jadwal yang optimum. Permasalahan penjadwalan pengajaran klasik didefinisikan sebagai berikut [1]:

Terdapat sejumlah m kelas c_1, ..., c_m, n guru t_1, ..., t_n, dan p periode 1, ..., p. Terdapat pula matriks integer non-negatif $R_{m \times n}$, yang disebut matriks requirements, dengan r_{ij} adalah jumlah pelajaran yang diberikan oleh guru t_j pada kelas c_i. Permasalahan penjadwalan adalah mengalokasikan kelas pada periode sedemikian sehingga tidak ada pengajar dan kelas yang dijadwalkan terjadi pada satu pelajaran pada saat yang sama.

Definisi klasik ini belum dapat memenuhi kebutuhan penjadwalan perkuliahan di Perguruan Tinggi seperti:

- seorang dosen terkadang hanya dapat mengajar pada jam-jam dan hari-hari tertentu.
- setiap matakuliah memiliki alokasi semester dimana matakuliah itu diajarkan, sehingga perlu pengaturan agar penjadwalan matakuliah pada semester yang sama tidak bersamaan agar mahasiswa pada semester tersebut dapat mengikuti semua matakuliah yang dialokasikan kepadanya.
- dalam satu hari seorang dosen seharusnya maksimal mengajar dua kelas.

Masalah penjadwalan perkuliahan di Perguruan Tinggi merupakan masalah *NP-hard* secara komputasi, sehingga sejumlah penelitian (seperti pada [2, 3, 4, 5]) menerapkan metode *heuristic* untuk melakukan otomasi terhadap masalah penjadwalan.

Masalah penjadwalan perkuliahan berbeda dari satu universitas ke universitas, bahkan dari satu jurusan ke jurusan lain pada universitas yang sama. Pada penelitian ini studi kasus yang dipilih adalah masalah penjadwalan pada jurusan penulis, yaitu Jurusan Teknik Informatika, Fakultas Teknologi Informasi, ITS. Pembuatan jadwal pada Jurusan ini harus dilakukan pada setiap pergantian semester. Padahal pembuatan jadwal ini membutuhkan waktu, tenaga dan ketelitian untuk membuatnya, karena itu diperlukan penjadwalan otomatis yang dapat membuat jadwal dengan cepat dan mudah, sehingga masalah pembuatan jadwal matakuliah dapat diselesaikan dengan lebih efisien. Dalam pembuatan jadwal tersebut harus memperhatikan aturan-aturan penjadwalan yang sudah ditentukan.

Pada penelitian ini metode Algoritma Genetika dipadukan dengan metode *Constraint Satisfaction Problem*, dimana kromosom yang dihasilkan pada metode Algoritma Genetika dapat diproses dengan metode *Constraint Satisfaction Problem* sehingga dapat ditemukan batasan-batasan pada penjadwalan yang harus dipenuhi dengan cepat dan akurat. Hal ini akan membuat proses *heuristic* pada Algoritma Genetika menjadi terarah dan membuat keseluruhan proses menjadi lebih efisien.

2 DASAR TEORI

Dasar teori yang digunakan dalam sistem ini meliputi *Constraint Satisfaction Problem* (CSP) dan Algoritma Genetika. CSP adalah suatu permasalahan dimana seseorang harus mencari nilai untuk set variabel (finite) yang memenuhi set *constraint (finite)* [6, 7]. CSP terdiri dari komponen-komponen berikut:

- Variabel, yang merupakan penampung yang dapat diisi berbagai nilai.
- *Constraint* yang merupakan suatu aturan yang ditentukan, yang akan mengatur nilai yang boleh diisikan ke variabel, atau kombinasi variabel.

- Domain yang merupakan kumpulan nilai legal yang dapat diisi ke variabel.
- Solusi yang merupakan *assignment* nilai-nilai dari domain ke setiap variabel dengan tidak ada *constraint* yang dilanggar.

CSP dimulai dari solusi kosong dan diakhiri dengan sebuah solusi yang memenuhi semua *constraint (consistent)*. Pencarian solusi dilakukan dengan mencoba mengisikan nilai dari *domain* kepada setiap variabel satu demi satu tanpa melanggar *constraint*, sampai solusi ditemukan.

Algoritma yang paling banyak dipakai untuk melakukan pencarian sistematik untuk menyelesaikan CSP adalah *backtracking*. Algoritma *backtracking search* (penelusuran kembali) adalah suatu bentuk algoritma *depth-first-search*. *Backtracking* dapat dilihat sebagaimana *searching* dalam *tree*, dimana setiap *node* mewakili *state* dan turunan dari setiap *node* mewakili ekstensi dari *state* tersebut.

Pada metode *backtracking*, variabel diisikan secara *sequential* selagi semua variabel relevan dengan *constraint* yang sudah diinisialisasi. Jika solusi partial melanggar *constraint*, *backtracking* melakukan langkah kembali ke solusi partial sebelumnya dan memilih nilai lain yang belum dicoba untuk variabel yang ingin diisikan, langkah tersebut berguna untuk menghindari eksplorasi lebih lanjut dari solusi partial yang salah. Keuntungan dari *backtracking* adalah pemeriksaan *consistency* hanya perlu dilakukan terhadap *assignment* yang terakhir dilakukan, karena pemeriksaan terhadap *assignment* yang sebelumnya sudah dilakukan sebelumnya.

Pada algoritma *backtracking*, teknik *look ahead* digunakan untuk meramalkan efek pemilihan variabel *branching* untuk mengevaluasi nilai-nilai variabel tersebut. *Forward checking* adalah salah satu cara untuk melakukan *look ahead*. *Forward checking* mencegah terjadinya konflik dengan melarang nilai-nilai yang belum diisikan ke variabel untuk dipakai. Ketika suatu nilai diisikan ke suatu variabel, nilai yang berada di *domain* dari variabel yang konflik dengan *assignment* tersebut akan dihilangkan dari *domain*.

Minimum Remaining Value (MRV) adalah suatu teknik yang dikembangkan untuk menangani masalah besarnya kemungkinan gagal pada pencarian menggunakan CSP. MRV berkerja dengan memilih variabel yang memiliki *domain* legal paling sedikit (memiliki kemungkinan untuk membuat suatu *dead-end* paling besar) untuk diisikan terlebih dulu.

Algoritma Genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika. Algoritma ini adalah simulasi dari proses evolusi Darwin dan operasi genetika atas kromosom.

Metode yang dapat disebut algoritma genetika adalah metode yang setidaknya memiliki komponen seperti populasi dari kromosom, seleksi berdasarkan *fitness, crossover* untuk memproduksi *offspring* baru, dan random mutasi pada *offspring* [8].

Seleksi bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit* [9, 8]. Seleksi akan menentukan individu-individu mana saja yang akan dipilih untuk mendapatkan generasi baru.

Crossover adalah proses dari pertukaran dua kromosom yang menciptakan *offspring* yang mewarisi karakter dari kedua parent. Dengan mewarisi karakter parent tersebut tujuan dari melakukan *crossover* adalah untuk mencari kombinasi gen terbaik dari parent.

Operator Mutasi mengubah field individual (*gen*) didalam kromosom berdasarkan pada probabilitas. Mutasi dipakai untuk menghindari kecenderungan suatu populasi untuk menjadi mirip [8].

Fitness adalah suatu nilai yang dipakai untuk tolak ukur kualitas kromosom [9]. Nilai *fitness* ini digunakan untuk menentukan kromosom yang berkualitas lebih baik daripada kromosom yang lain.

Algoritma Genetika memiliki algoritma umum dalam mencari solusi melalui pembangkitan populasi kromosom, mekanisme seleksi berdasarkan *fitness, crossover* untuk memproduksi *offspring* baru, dan mutasi acak pada *offspring*. Namun algoritma tersebut perlu penyesuaian untuk permasalahan yang dihadapi. Pada penelitian ini Algoritma Genetika digunakan karena memiliki keunggulan utama yaitu efisien waktu dalam pencarian solusi. Namun algoritma ini juga memiliki kekurangan utama yaitu tidak adanya jaminan solusi yang dihasilkan adalah solusi terbaik (bila dibandingkan dengan pencarian melalui evaluasi lengkap pada semua kemungkinan solusi).

3 PERANCANGAN SISTEM

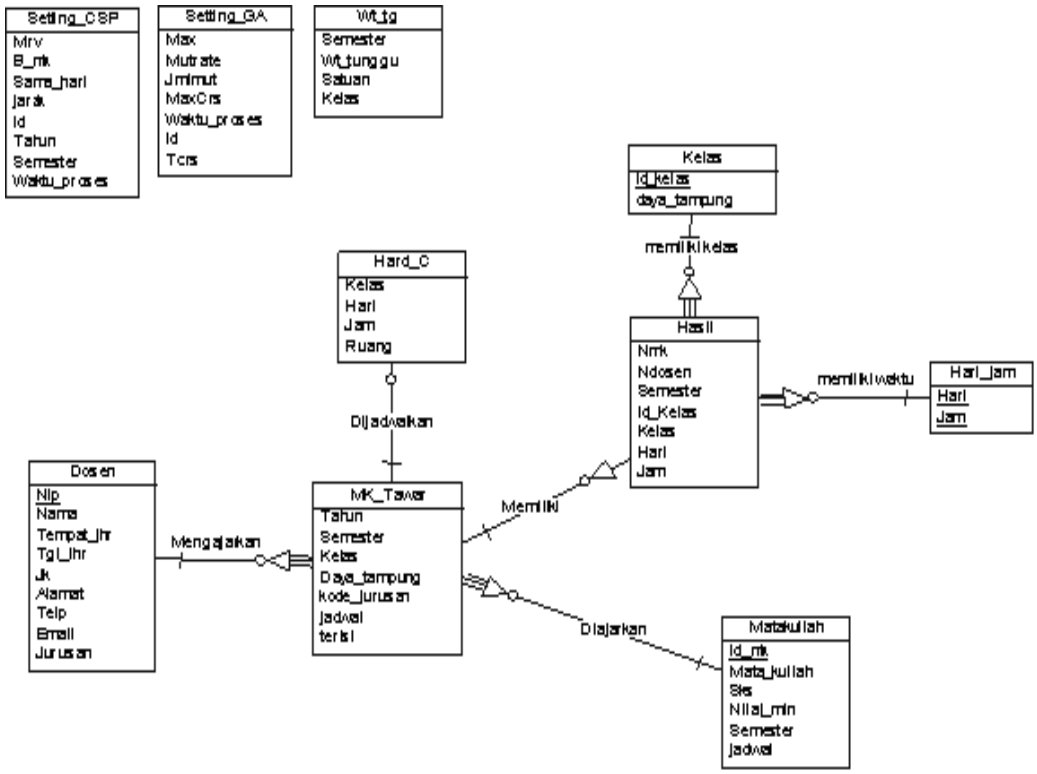
Seperti yang telah dituliskan pada Pendahuluan, studi kasus yang diambil pada penelitian ini adalah pada jurusan penulis, yaitu Jurusan Teknik Informatika, Fakultas Teknologi Informasi, ITS. Hasil (*output*) uji coba dari aplikasi diverifikasi secara manual untuk membuktikan tidak ada jadwal yang melanggar aturan yang telah ditetapkan.

Agar suatu jadwal dapat dibuat dengan benar, terdapat sejumlah faktor dan aturan penjadwalan harus diperhatikan. Faktor-faktor yang berpengaruh dalam pembentukan jadwal meliputi:

1. Dosen
   Seorang dosen tidak dapat mengajar beberapa mata kuliah pada jam yang sama. Selain itu, seorang dosen terkadang hanya dapat mengajar pada jam-jam dan hari-hari tertentu saja, sehingga perlu untuk memesan jadwal khusus yang tidak dapat diganggu mata kuliah yang lain.

2. Ruang
   Mengingat jumlah ruang yang dimiliki terbatas, maka perlu diperhatikan ruang yang tersedia agar tidak menggangu jalannya perkuliahan. Jadwal harus hanya mengakomodir ruang yang ada.

3. Waktu
   Waktu merupakan batasan berapa menit yang diperlukan untuk satu jam kuliah. Selain itu ada hari-hari dimana jam kuliah dibatasi sampai jam tertentu (misalnya pada hari Jumat jam kuliah dibatasi mulai jam 07.30 sampai jam 11.00 dan dimulai kembali jam 13.00). Dengan batasan-batasan waktu ini, jadwal hanya akan berada pada waktu yang ditentukan.



Gambar 1: ERD Sistem Penjadwalan

4. Matakuliah
   Mengingat setiap matakuliah memiliki semester mata kuliah itu diajarkan. Maka perlu adanya aturan yang membatasi penjadwalan matakuliah, agar matakuliah itu sesuai dengan aturan-aturan penjadwalan.

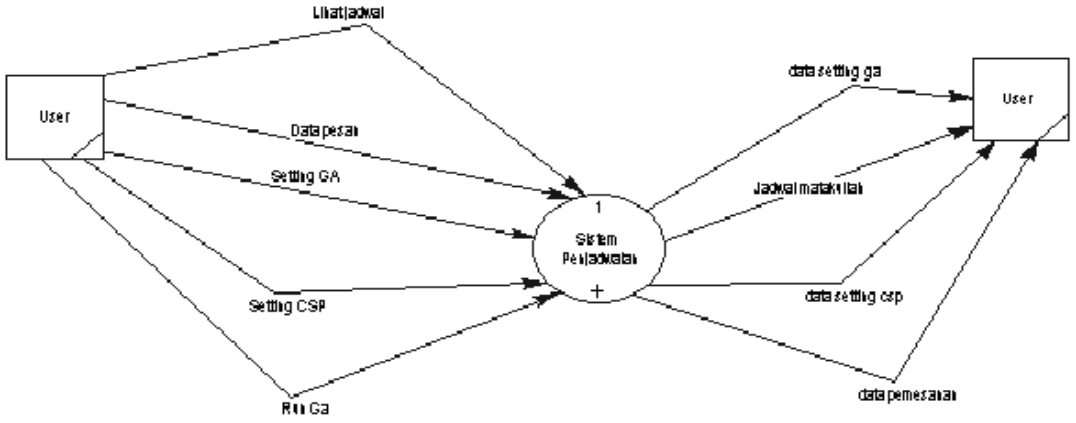
Aturan-aturan yang harus diperhatikan dalam membuat jadwal meliputi:

- Tidak boleh ada satu ruangan yang diisikan dua kali dalam satu waktu yang sama.
- Tidak boleh ada Nomor Induk Pegawai (NIP) dosen yang sama pada hari dan jam yang sama.
- Kemunculan matakuliah pada semester yang sama dibatasi maksimal dua kali pada satu hari.
- Mahasiswa pada masing-masing semester harus dapat mengambil matakuliah sesuai dengan kurikulum yang ditentukan jurusan, sehingga jika mahasiswa mengambil matakuliah sesuai kurikulum maka tidak ada matakuliah yang tidak bisa diambil karena jadwal yang bersamaan dengan jadwal matakuliah lain pada semester yang sama.
- Matakuliah dengan kode matakuliah yang sama (kelas paralel) harus ada pada hari yang sama, kecuali jika dosennya sama.
- Jarak matakuliah antar semester dengan semester berikutnya pada jam yang sama harus lebih besar dari jarak yang ditetapkan pengguna. Ini berguna untuk memberikan keleluasaan mahasiswa mengambil matakuliah yang tidak berada pada semester mahasiswa tersebut, misalnya dalam kasus mahasiswa mengulang suatu matakuliah atau mahasiswa yang memiliki Indeks Prestasi tinggi mengambil suatu matakuliah lain di semester lebih tinggi.
- Dalam satu hari, dosen mengajar maksimal dua kelas.

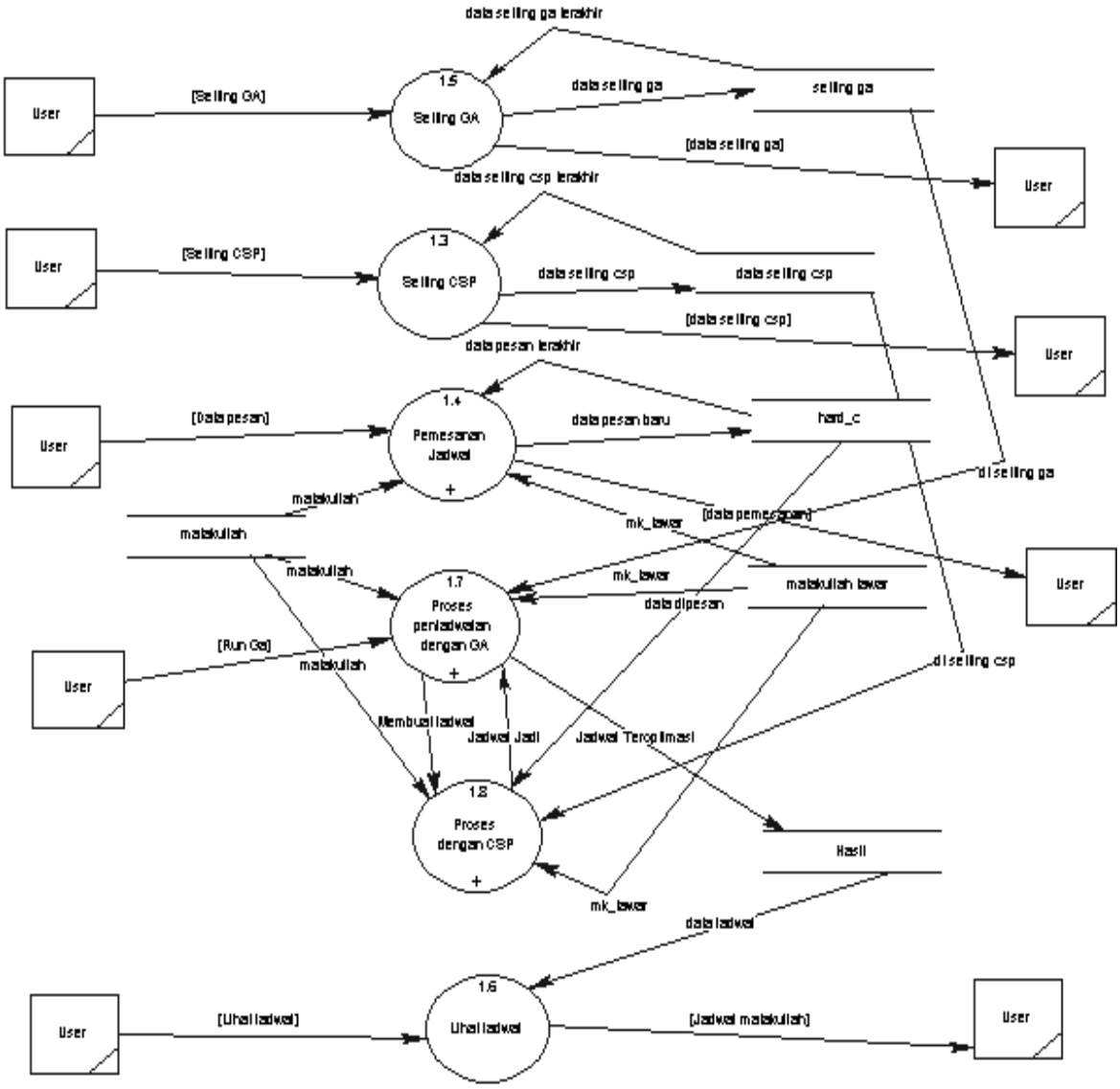
Untuk penyimpanan pada sistem digunakan data master yang tersimpan pada *database*. Data ini merupakan data untuk membuat penjadwalan yang diambil dari FRS-online Jurusan Teknik Informatika ITS (entitas-entitas yang diambil adalah entitas dosen, entitas matakuliah, dan entitas matakuliah yang ditawarkan). Data untuk inisialisasi Algoritma Genetika dan *Constraint Satisfaction Problem* meliputi data *setting_csp*, *setting_ga, wt_tg, kelas, hari_jam, hasil, hard_c*. Hubungan antar entitas ditunjukkan dengan *Entity Relationship Diagram* (ERD) pada Gambar 1.

Data proses adalah data yang digunakan selama proses pembuatan jadwal. Data ini diolah dari data masukan dan digunakan untuk menghasilkan data keluaran. Adapun data utama pada proses yaitu Pemrosesan Algoritma Genetika, dan Pemrosesan *Constraint Satisfaction*.

Data keluaran adalah informasi yang dihasilkan oleh sistem untuk pengguna. Pada tahapan ini akan diperoleh jadwal yang telah dioptimasi terhadap waktu tunggu antar kuliah mahasiswa. Jadwal yang dihasilkan tersebut merupakan hasil yang dianggap paling baik setelah proses selesai. Dengan kata lain jadwal yang dihasilkan memiliki



Gambar 2: DFD Level 0 Sistem Penjadwalan



Gambar 3: DFD Level 1 Sistem Penjadwalan

kromosom-kromosom yang memiliki nilai *fitness* paling baik.

Gambar 2 menunjukkan hubungan dan aliran data dalam bentuk *Data-Flow Diagram* (DFD)level 0 dari sistem yang dirancang.

Dekomposisi terhadap DFD level 0 menghasilkan DFD level 1 seperti ditunjukkan pada Gambar 3. Terdapat enam proses utama untuk sistem penjadwalan, yaitu: proses *Setting CSP*, proses *Setting GA*, proses *Pemesanan Jadwal*, proses *Penjadwalan Menggunakan GA*, proses *Penjadwalan dengan CSP*, dan proses *Lihat Jadwal*.

Fungsi masing-masing proses pada Gambar 3 adalah sebagai berikut:

1. Proses *Setting GA*.
   Proses ini melakukan perubahan nilai *setting* Algoritma Genetika agar algoritma berjalan sesuai yang diharapkan oleh pengguna.

2. Proses *Setting CSP*.
   Proses ini melakukan perubahan nilai *setting* atau *constraint* pada CSP, agar jadwal yang terbentuk sesuai dengan keinginan pengguna. Jika dalam *setting* terjadi perubahan tahun atau semester maka secara langsung data pemesanan jadwal pada *database* akan dihapus.

3. Proses *Pemesanan Jadwal*.
   Proses ini melakukan pemilihan terhadap jadwal matakuliah yang memesan waktu dan tempat tertentu

4. Proses *Penjadwalan Menggunakan GA*.
   Proses ini melakukan proses pembuatan jadwal dengan menggunakan Algoritma Genetika sebagai pengatur arah pencarian nilai yang optimal.

5. Proses *Penjadwalan dengan CSP*.
   Proses ini melakukan proses pembuatan jadwal dengan menggunakan CSP.

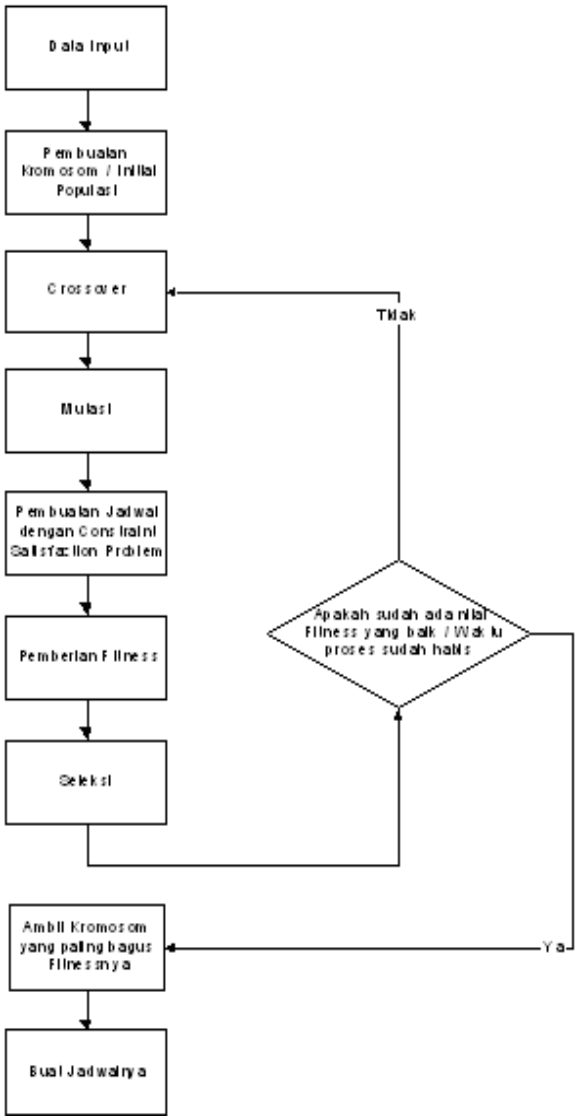
6. Proses *Lihat Jadwal*.
   Proses ini menampilkan jadwal yang terbentuk.

Tahapan pada proses Penjadwalan Menggunakan GA dimulai dengan mempersiapkan tabel hasil. Jika tabel hasil sudah diinisialisasi maka prosedur Algoritma Genetika akan dimulai, selanjutnya terjadi pembentukan kromosom pada Algoritma Genetika. Kromosom dibentuk berdasarkan data kromosom yang datanya tersimpan pada *database*. Setelah proses inisialisasi selesai maka proses mengambil *setting* GA untuk mengatur jalannya Algoritma Genetika sehingga sesuai dengan kebutuhan pengguna. Setelah itu Algoritma Genetika dijalankan untuk mendapatkan jadwal matakuliah yang optimal. Gambar 4 menunjukkan *flowchart* proses yang dilakukan Algoritma Genetika.

Proses yang terjadi dalam Algoritma Genetika adalah seperti yang terlihat dalam Gambar 4, yaitu:

1. Melakukan pengambilan data yang dibutuhkan Algoritma Genetika.

2. Melakukan pemilihan kromosom untuk dilakukan *crossover* dan mutasi. Hasilnya kemudian ditambahkan ke dalam populasi.



Gambar 4: *Flowchart* Proses Algoritma Genetika

3. Melakukan pembuatan jadwal dari setiap data kromosom yang baru, dengan menggunakan *constraint satisfaction*.

4. Mencari nilai waktu tunggu antar kuliah rata-rata mahasiswa dari setiap data baru.

5. Melakukan seleksi, jika nilai *fitness* yang yang dicari belum tercapai dan waktu belum habis kembali ke langkah 2.

6. Mengambil kromosom dengan nilai *fitness* terbaik sebagai solusi dan buat jadwalnya.

Hal-hal yang dilakukan pada proses inisialisasi Algoritma Genetika adalah:

1. Menginisialisasi tabel hasil, dengan menghapus data yang lama dan mengisikan data default.

2. Mengambil data *setting* paramater Algoritma Genetika dari *database*.

3. Mengambil data matakuliah yang ditawarkan dan matakuliah dari *database* sebagai data kromosom.

4. Melakukan pembentukan kromosom sebagai populasi awal.

5. Melakukan proses *preprocessing* untuk memperkirakan apakah jadwal dapat dibuat.
   Proses ini melakukan penghitungan jumlah data yang dibutuhkan untuk membuat jadwal dan membandingkannya dengan jumlah *domain* yang tersedia. Jika *domain* tersebut mencukupi maka proses pembuatan jadwal dapat dilanjutkan, sedangkan jika *domain* tidak mencukupi maka jadwal dianggap tidak dapat dibuat dan proses dibatalkan.

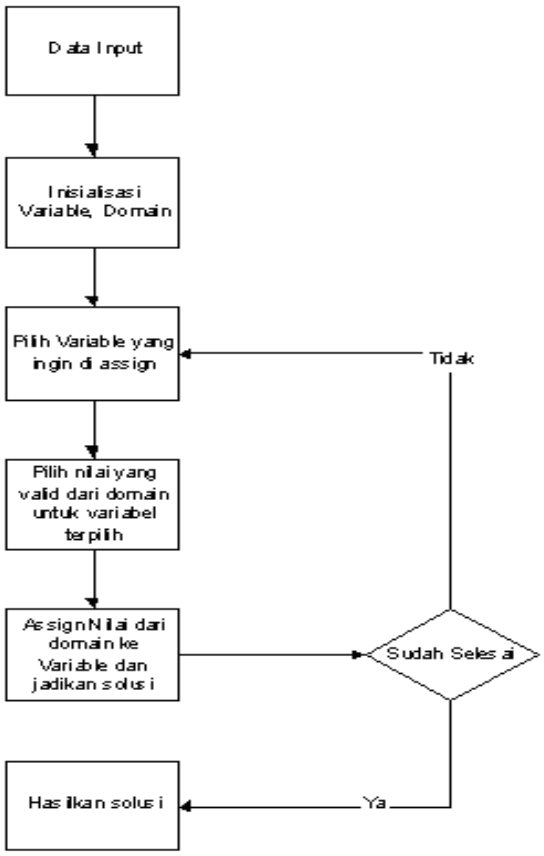
Model Algoritma Genetika yang akan digunakan untuk melakukan optimasi adalah sebagai berikut:

1. Seleksi.
   Pada seleksi, dilakukan penilaian atas nilai *fitness*. Sehingga *fitness* yang memiliki kualitas kromosom paling baik memiliki kemungkinan terpilih ke dalam generasi selanjutnya lebih besar. Seleksi yang dipakai disini adalah seleksi yang menggunakan metode *roulette wheel*. Pada seleksi ini perlu diperhatikan jumlah maksimal populasi sebagai input, agar populasi tidak menjadi terlalu besar dan memakan banyak waktu, dan agar populasi juga tidak menjadi terlalu kecil dan mengakibatkan kromosom terlalu mirip sehingga operasi kromosom tidak akan banyak berpengaruh.

2. *Crossover*.
   Crossover yang digunakan adalah penyilangan dua titik dengan permutasi. Pemilihan kromosom yang akan di-*crossover*-kan ditentukan oleh probabilitas yang ditentukan dalam *setting* parameter Algoritma Genetika. Dan banyaknya *gen* yang ditukar juga mengikuti *setting* parameter Algoritma Genetika. Dalam melakukan penyilangan, setiap dua kromosom akan menghasilkan dua *offspring* baru hasil penyilangan.

3. Mutasi.
   Mutasi dilakukan setelah operasi kromosom ini dilakukan dengan menukar gen secara random. Dalam proses ini perlu diperhatikan tingkat mutasi dan tingkat probabilitas terjadi mutasi. Jika sering terjadi mutasi, maka antar suatu generasi ke generasi selanjutnya akan kehilangan kemiripan dan pencarian akan menjadi acak. Tetapi jika mutasi terlalu sedikit maka kromosom akan menjadi terlalu mirip dan kromosom baru akan muncul terlalu lama pada populasi.

4. Penentuan *Fitness*.
   Penentuan *fitness* pada dasarnya adalah pemberian nilai kuantitatif yang mewakili kualitas dari kromosom. Hal pertama yang dilakukan proses ini adalah



Gambar 5: *Flowchart* Pembuatan Jadwal dengan Menggunakan CSP

proses Pembuatan Jadwal sesuai kromosom yang dipilih, dengan memprosesnya dengan *Constraint Satisfaction Problem*, dan dari hasilnya akan diberi nilai fitness.
Fungsi *fitness* yang dibuat merupakan rata-rata waktu tunggu antar kuliah mahasiswa dalam satu minggu, sehingga semakin kecil waktu tunggu maka akan semakin baik (waktu tunggu antar kuliah mahasiswa menjadi minimal). Rumusan fungsi *fitness* $f(fit)$ ditunjukkan pada persamaan (1)

$$f(fit) = \frac{\sum_{i=1}^{n} \frac{R(i)}{n} + \sum_{j=1}^{m} \frac{X(j)}{m}}{2} \qquad (1)$$

dengan $R(i)$ adalah waktu tunggu antar kuliah mahasiswa reguler dalam satu hari, dan $X(j)$ adalah waktu tunggu antar kuliah mahasiswa ekstensi dalam satu hari. Nilai n adalah jumlah mahasiswa reguler dalam satu minggu, dan m adalah jumlah mahasiswa ekstensi dalam satu minggu.

Adapun proses yang terjadi pada proses *Penjadwalan dengan CSP* dijelaskan pada uraian berikut. Proses pertama yang dilakukan adalah mengambil data *setting* CSP yang dipakai untuk mengatur pembentukan jadwal. Setelah itu dilakukan inisialisasi *Constraint Satisfaction Problem*, yaitu pembentukan solusi awal yang dibentuk dari

Tabel 1: Pemesanan Matakuliah pada Semester Genap Tahun Ajaran 2006/2007

Id_mk	NIP	SMT	Kelas	Hari	Jam	Ruang
CI1307	130816212	2	A	4	1	1
CI1504	051100002	3	XRA	5	4	1
CI1205	131996151	8	B	3	1	1
CI1422	51100013	6	XA	5	5	1

data pesan jadwal yang sudah dibuat. Setelah data pesan jadwal valid maka dilakukan pembuatan jadwal menggunakan CSP. Gambar 5 menunjukkan *flowchart* pembuatan jadwal dengan menggunakan CSP yang dilakukan.
Proses yang terjadi dalam pembuatan jadwal pada Gambar 5 dapat diuraikan sebagai berikut:

1. Melakukan inisialisas, yaitu persiapan data yang dibutuhan oleh *Constraint Satisfaction*.
2. Memilih variabel yang ingin diisikan, jika melakukan pemilihan dengan menggunakan MRV maka diambil variabel yang memiliki *domain* paling sedikit, jika tidak menggunakan MRV akan diambil secara berurutan sesuai urutan pada variabel.
3. Memilih nilai yang masih valid dari *domain* untuk variabel yang dipilih.
4. Melakukan *assignment* dari variabel dan domain ke dalam solusi, kemudian mengurangi *domain* valid dari setiap variabel berdasarkan *constraint* yang dimiliki variabel yang di-*assign*.
5. Jika solusi sudah terbentuk maka lanjutkan ke langkah 6. Jika solusi belum ketemu maka kembali ke langkah 2.
6. Jika setiap variabel sudah terisi maka solusi sudah terbentuk, kemudian CSP akan memberikan hasilnya kembali ke Algoritma Genetika untuk dievaluasi.

Hal-hal yang dilakukan pada proses inisialisasi CSP adalah:

1. Mengambil *setting* CSP dari database untuk menentukan *constraint* dan metode yang dipakai.
2. Mengalokasikan data matakuliah yang ditawarkan (yang diberikan dari Algoritma Genetika) menjadi variabel yang harus diselesaikan.
3. Mengambil data *hard_c* untuk inisialisasi variabel yang dipesan terlebih dahulu sebelum melakukan penjadwalan pada variabel yang lain.

Model *Constraint Satisfaction* yang digunakan untuk menyelesaikan masalah penjadwalan adalah sebagai berikut:

1. Penelusuran dengan *Backtracking* (BT).
   Algoritma *backtracking* menggunakan *Constructive methods*, yang berarti mengembangkan solusi *partial* sedikit demi sedikit dengan tetap menjaga *consistency*, sehingga mencapai *consistent complete assignment*.

Tabel 2: Hasil Pembuatan Jadwal

Idmk	Kel	NDosen	Hari	Jam	idkel
CI1205	A	Joko Lianto B	1	1	1
CI1305	C	Nunut Priyo J	1	1	2
CI1305	B	F.X. Arunanto B	1	1	3
CI1305	A	Nanik Suciati B	1	1	4
CI1421	B	Ahmad Khoirul Bashori	1	1	5
CI1421	A	Daniel O. Siahaan	1	1	6
CI1307	B	Esther Hanaya	1	2	1
CI1412	C	Ary Mazharuddin Shiddiqi	1	2	2
CI1412	A	Royyana Muslim I	1	2	3
CI1412	B	Muchammad Husni	1	2	4
CI1424	A	Dwi Sunaryono	1	2	5
CI1414	A	Victor Hariadi	1	3	1
CI1414	C	Yudhi Purwananto	1	3	2
CI1414	B	Bilqis Amaliah	1	3	3
CI1424	B	Dwi Sunaryono	1	3	4
CI1501	XRA	Handayani Tjandrasa	1	4	1
CI1502	XRA	Nanik Suciati	1	4	2
CI1423	XA	Chastine Fatichah	1	5	1
CI1205	XA	Joko Lianto B	1	5	2
CI1202	A	Nunut Priyo J	2	1	1
CI1202	B	Ary Mazharuddin Shiddiqi	2	1	2
CI1409	A	Darlis Heru Murti	2	1	3
CI1423	A	Rully Soelaiman	2	1	4
CI1202	C	Ary Mazharuddin Shiddiqi	2	2	1
CI1409	B	Darlis Heru Murti	2	2	2
CI1423	B	Rully Soelaiman	2	2	3
CI1203	B	Arif Bramantoro	2	3	1
CI1203	A	Dwi Sunaryono	2	3	2
CI1203	C	Ahmad Khoirul Bashori	2	3	3
CI1422	B	Siti Rochimah	2	3	4
CI1422	A	Isye Ariesanti	2	3	5
CI1512	XRA	Waskitho Wibisono	2	4	1
CI1516	XRA	FX.Arunanto	2	4	2
CI1421	XA	Ahmad Khoirul Bashori	2	5	1
CI1205	B	Joko Lianto B	3	1	1
CI1409	C	Darlis Heru Murti	3	1	2
CI1422	C	Siti Rochimah	3	1	3
CI1410	A	Imam Kuswardayan	3	2	1
CI1410	B	Umi Laili Yuhana	3	2	2
CI1410	C	Umi Laili Yuhana	3	3	1
CI1521	XRA	Suhadi Lili	3	4	1
CI1523	XRA	Riyanarto Sarno	3	4	2
CI1424	XA	Imam Kuswardayan	3	5	1
CI1307	A	Esther Hanaya	4	1	1
CI1307	C	Chastine Fatichah	4	1	2
CI1411	A	Arif Bramantoro	4	1	3
CI1411	B	Fajar Baskoro	4	1	4
CI1411	C	Fajar Baskoro	4	2	1
CI1522	XRA	Suhadi Lili	4	4	1
CI1513	XRA	Wahyu Suadi	4	4	2
CI1504	XRA	Irfan Subakti	5	4	1
CI1422	XA	Isye Ariesanti	5	5	1

2. *Forward Checking* (FC).
   Forward checking berkerja dengan membuat suatu tabel yang menyatakan nilai *domain* yang masih tersedia untuk setiap variabel. Kemudian pada tabel ini dihilangkan *domain* yang sudah tidak boleh diambil lagi.
3. *Minimum Remaining Value* (MRV).
   Minimum remaining value dapat bekerja bersamaan dengan FC dengan memilihkan variabel yang diisikan. Variabel yang dipilih adalah variabel yang memiliki *domain* paling sedikit. Pertimbangan yang digunakan adalah variabel yang jumlah *domain*-nya paling sedikit mempunyai kesempatan gagal lebih besar. Hal ini dapat memotong percabangan *tree*

yang tidak perlu dan mempercepat waktu pembuatan jadwal.

Pembuatan sistem penjadwalan ini dilakukan dengan menggunakan ASP untuk pembuatan antarmuka dan Oracle 9.2.0.1.2 untuk pembuatan proses penjadwalan. Pembuatan dengan ASP meliputi (i)form halaman depan, (ii)form setting GA, (iii)form setting CSP, (iv)form pemesanan jadwal, dan (v)form pembuatan jadwal. Sedangkan pembuatan dengan Oracle meliputi implementasi proses Algoritma genetika dan implementasi proses *Constraint Satisfaction Problem*.

4 UJI COBA

Pada tahap uji coba dilakukan dengan tiga skenario yang ditujukan untuk mengetahui fungsionalitas sistem yang dibuat. Uji coba ini dilakukan dengan menggunakan lingkungan dengan spesifikasi sebagai berikut:

Spesifikasi perangkat lunak:

- Microsoft Windows server 2003
- Oracle 9.2.0.2.1
- Quest Software, Toad 7.6
- Microsoft Visual Studio.NET 2003
- ASP.NET

Spesifikasi perangkat keras:

- Athlon XP 1.3 MHz
- RAM 384 MB
- 40 GB Harddisk

Sebelum uji coba dilakukan, terlebih dahulu dipersiapkan data pada aplikasi (dari *database* FRS *online* ITS untuk mahasiswa Jurusan Teknik Informatika) yaitu data kelas yang ditawarkan. Ada tiga skenario yang akan diuji.

4.1 Skenario 1

Skenario ini dijalankan untuk menguji kemampuan *constraint satisfaction* dalam membuat jadwal serta menangani pemesanan jadwal pada waktu dan kelas tertentu. Dalam skenario ini Algoritma Genetika belum difungsikan sehingga kemampuan *constraint satisfaction* dari sistem yang dibuat dapat dibuktikan kebenarannya. Untuk skenario ini diambil pemesanan matakuliah pada semester genap tahun ajaran 2006/2007 yang sebagian datanya ditunjukkan pada Tabel 1.

Constraint utama yang digunakan adalah sebagai berikut:

- Tidak ada satu ruangan yang diisikan dua kali dalam satu waktu yang sama.
- Tidak ada NIP yang sama pada hari dan jam yang sama.
- Tidak ada semester yang sama pada waktu yang sama.

Constraint tambahan yang digunakan adalah sebagai berikut:

- Membatasi matakuliah pada semester yang sama hanya muncul dua kali pada satu hari.
- Matakuliah dengan kode matakuliah yang sama harus ada pada hari yang sama, kecuali jika dosennya sama.
- Jarak antar semester dengan semester berikutnya pada jam yang sama harus lebih besar dari jarak yang diinputkan.
- Dosen yang sama dalam satu hari hanya mengajar dua kelas.

Tabel 2 menunjukkan jadwal yang dihasilkan oleh sistem. Waktu proses yang diperlukan untuk menghasilkan jadwal tersebut adalah sekitar 50 detik.

Verifikasi secara manual terhadap hasil pembuatan jadwal pada Tabel 2 menunjukkan bahwa tidak ada jadwal yang menyalahi *constraint* yang diberikan, sehingga disimpulkan kemampuan *constraint satisfaction* pada sistem telah bekerja sebagaimana seharusnya.

4.2 Skenario 2

Skenario ini digunakan untuk menguji kemampuan Algoritma Genetika untuk mendapatkan jadwal dengan waktu tunggu antar kuliah mahasiswa. *Setting* parameter Algoritma Genetika yang digunakan adalah sebagai berikut:

- Populasi maksimum: 10
- Kemungkinan terjadinya crossover: 30
- Kemungkinan terjadinya mutasi: 2
- Jumlah data yang disilangkan: 50
- Nilai fitness yang dituju: 0 (berarti menuju keadaan ideal yaitu tidak ada waktu tunggu antar kuliah mahasiswa).

Tabel 3 menunjukkan statistik *fitness* tiap generasi. Waktu yang diperlukan untuk menghasilkan jadwal tersebut adalah sekitar 45 detik.

Dari Tabel 3 terlihat bahwa waktu tunggu mahasiswa (ditunjukkan dengan nilai *fitness*) semakin lama semakin kecil. Hal ini menunjukkan optimasi pada Algoritma Genetika pada sistem telah bekerja sebagaimana seharusnya.

4.3 Skenario 3

Skenario ini juga dimaksudkan untuk menguji kemampuan Algoritma Genetika untuk mendapatkan jadwal dengan waktu tunggu antar kuliah mahasiswa seperti Skenario 2, namun dengan *setting* parameter Algoritma Genetika yang berbeda yaitu:

- Maksimum populasi : 10
- Kemungkinan terjadinya crossover : 30
- Kemungkinan terjadinya mutasi : 20
- Jumlah data yang disilangkan : 50

Tabel 3: Statistik Fitness Setiap Generasi untuk Skenario 2

Generasi	Fitness Terbaik	Fitness Terburuk	Fitness Rata-Rata
1	0,501190476	0,501190476	0,501190476
2	0,501190476	0,501190476	0,501190476
3	0,501190476	0,501190476	0,501190476
4	0,501190476	0,501190476	0,501190476
5	0,501190476	0,501190476	0,501190476
6	0,501190476	0,601190476	0,53452381
7	0,501190476	0,601190476	0,53452381
8	0,476190476	0,601190476	0,528452381
9	0,476190476	0,601190476	0,513452381
10	0,43452381	0,601190476	0,494285714
11	0,419642857	0,648809524	0,490892857
12	0,392857143	0,648809524	0,485059524
13	0,392857143	0,501190476	0,448035714
14	0,392857143	0,601190476	0,455357143
15	0,392857143	0,601190476	0,44702381
16	0,392857143	0,476190476	0,426190476
17	0,392857143	0,476190476	0,426190476
18	0,392857143	0,476190476	0,417857143
19	0,392857143	0,392857143	0,392857143

- Waktu maksimal pelaksanaan Algoritma
- Genetika dibatasi maksimal 300 detik
- Nilai fitness yang dicari : 0
  (berarti menuju keadaan ideal yaitu tidak ada waktu tunggu antar kuliah mahasiswa).

Selain itu pada skenario ini ditetapkan *constraint* tambahan sebagai berikut:

- Membatasi matakuliah pada semester yang sama hanya muncul dua kali pada satu hari.
- Matakuliah dengan kode matakuliah yang sama harus ada pada hari yang sama
- Jarak antar semester dengan semester berikutnya pada jam yang sama harus lebih besar dari jarak yang diinputkan.

Tabel 4 menunjukkan statistik *fitness* tiap generasi yang dihasilkan.

Dari Tabel 4 juga terlihat bahwa waktu tunggu mahasiswa semakin lama semakin kecil. Hal ini menunjukkan bahwa dengan nilai *setting* parameter yang berbeda, optimasi pada Algoritma Genetika pada sistem juga telah bekerja sebagaimana seharusnya.

5 KESIMPULAN

Berdasarkan analisis hasil uji coba sistem maka dapat diambil kesimpulan-kesimpulan berikut:

1. Melalui percobaan skenario 1 dengan mengolah jadwal pada semester genap tahun ajaran 2006/2007, sistem telah mampu menangani pemesanan jadwal pada waktu tertentu, telah mampu mengolah data matakuliah yang ditawarkan, dan telah mampu menghasilkan jadwal tanpa ada *constraint* yang terlanggar.

Tabel 4: Statistik Fitness Setiap Generasi untuk Skenario 3

Generasi	Fitness Terbaik	Fitness Terburuk	Fitness Rata-Rata
1	0,375	0,375	0,375
2	0,375	0,375	0,375
3	0,375	0,375	0,375
4	0,375	0,375	0,375
5	0,375	0,375	0,375
6	0,375	0,375	0,375
7	0,375	0,375	0,375
8	0,375	0,375	0,375
9	0,375	0,375	0,375
10	0,375	0,375	0,375
11	0,375	0,402777778	0,388888889
12	0,361111111	0,402777778	0,383333333
13	0,208333333	0,402777778	0,362847222
14	0,208333333	0,516666667	0,383611111
15	0,208333333	0,572222222	0,391944444
16	0,208333333	0,572222222	0,391944444
17	0,208333333	0,516666667	0,363333333
18	0,208333333	0,516666667	0,360555556
19	0,208333333	0,405555556	0,346666667
20	0,208333333	0,405555556	0,324722222
21	0,180555556	0,488888889	0,296944444
22	0,180555556	0,488888889	0,297777778

2. Melalui pengamatan statistik yang dihasilkan dari percobaan skenario 2 terlihat bahwa Algoritma Genetika pada sistem telah melakukan optimasi dalam hal mencari waktu tunggu antar kuliah mahasiswa yang minimal.
3. Studi kasus yang diambil pada penelitian ini adalah pada jurusan penulis, yaitu Jurusan Teknik Informatika, Fakultas Teknologi Informasi, ITS. Aplikasi yang dibuat dapat diterapkan pada masalah penjadwalan matakuliah di Jurusan lain dengan menyesuaikan data masukan beserta *constraint* yang berlaku pada Jurusan tersebut.

DAFTAR PUSTAKA

[1] Werra, D.D.: *An Introduction to Timetabling*. European Journal of Operational Research **19**(2) (1985) 151–162

[2] Arntzen, H., Løkketangen, A.: *A Local Search Heuristic for a University Timetabling Problem*. Technical report, Dalle Molle Institute for Artificial Intelligence (2007) Diakses 31 August 2007, `http://www.idsia.ch/Files/ttcomp2002/arntzen.pdf`.

[3] Chiarandini, M., Birattari, M., Socha, K., Rossi-Doria, O.: *An Effective Hybrid Algorithm for University Course Timetabling*. Journal of Scheduling **9**(5) (2006) 403–432

[4] Corr, P.H., McCollum, B., McGreevy, M.A.J., McMullan, P.J.P.: *A New Neural Network-based Construction Heuristic for the Examination Timetabling Problem*. In: Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, LNCS 4193. (2006) 392–401

[5] Ozcan, E., Alken, A.: *Timetabling using a Steady State Genetic Algorithm*. In: The 4th international conference on the Practice And Theory of Automated Timetabling. (2002)

[6] Barták, R.: *On-Line Guide To Constraint Programming*. http://kti.mff.cuni.cz/~bartak/constraints/ (2007)

[7] Madsen, J.N.: *Methods for Interactive Constraint Satisfaction*. Master's thesis, Department of Computer Science, University of Copenhagen (2003)

[8] Mitchell, M.: *An Introduction to Genetic Algorithms*. The MIT Press (1999)

[9] Kusumadewi, S., Purnomo, H.: *Penyelesaian Masalah Optimasi dengan Teknik-teknik Heuristik*. Graha Ilmu (2005)